计算机科学
COMPUTER SCIENCE

融合麻雀搜索和随机差分的双向学习平衡优化器算法

侯新宇, 鲁海燕, 卢梦蝶, 徐杰, 赵金金

引用本文

侯新宇, 鲁海燕, 卢梦蝶, 徐杰, 赵金金. 融合麻雀搜索和随机差分的双向学习平衡优化器算法[J]. 计算机科学, 2023, 50(11): 248-258.

HOU Xinyu, LU Haiyan, LU Mengdie, XU Jie, ZHAO Jinjin. Bidirectional Learning Equilibrium Optimizer Combining Sparrow Search and Random Difference [J]. Computer Science, 2023, 50(11): 248-258.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221100143

融合麻雀搜索和随机差分的双向学习平衡优化器算法

侯新宇[1] 鲁海燕[1,2] 卢梦蝶[1] 徐 杰[1] 赵金金[1]
1 江南大学理学院 江苏 无锡 214122
2 无锡市生物计算工程技术研究中心 江苏 无锡 214122
(hxyu2016@sohu.com)

摘 要 针对平衡优化器算法(Equilibrium Optimizer,EO)求解精度低、收敛速度慢等问题,提出一种融合麻雀搜索和随机差分的双向学习平衡优化器算法。首先,给出了基于麻雀搜索算法的自适应种群划分策略,以平衡算法的全局探索和局部勘探,从而提高算法的收敛精度和收敛速度。其次,引入随机差分策略来重建平衡池,增加个体之间的信息交流,以利于算法跳出局部最优。最后,设计了一种双向混沌反向学习策略并将其应用到更新后的种群,以增加种群多样性,从而进一步提高算法的收敛精度。通过14个测试函数进行仿真实验,使用Wilcoxon秩和检验以及平均绝对误差来评价算法性能,并将改进算法应用到两个工程设计问题,实验结果验证了3种改进策略的有效性,且改进算法的收敛精度、收敛速度和鲁棒性都有显著提高。

关键词:平衡优化器算法;双向混沌反向学习;算法融合;随机差分;群智能优化算法
中图法分类号 TP301

Bidirectional Learning Equilibrium Optimizer Combining Sparrow Search and Random Difference

HOU Xinyu[1],LU Haiyan[1,2],LU Mengdie[1],XU Jie[1] and ZHAO Jinjin[1]
1 College of Science,Jiangnan University,Wuxi,Jiangsu 214122,China
2 Wuxi Biological Computing Engineering Technology Research Center,Wuxi,Jiangsu 214122,China

Abstract To address the problems of low solution accuracy and slow convergence speed of equilibrium optimizer,a bidirectional learning equilibrium optimizer combining sparrow search and random difference is presented. Firstly,an adaptive population division strategy based on sparrow search algorithm is proposed to balance the global exploration and local exploitation of the algorithm,so as to improve the convergence accuracy and convergence speed of the algorithm. Secondly,a random difference strategy is introduced to reconstruct the equilibrium pool and to increase the information exchange between individuals,so as to facilitate the algorithm to jump out of the local optimum. Finally,a bidirectional chaotic opposition learning strategy is designed and applied to the updated population to increase the population diversity and hence to further improve the convergence accuracy of the algorithm. Simulation experiments are conducted with 14 test functions,the performance of algorithm is evaluated using Wilcoxon rank-sum test and mean absolute error,and the improved algorithm is applied to two engineering design problems. Experimental results show that the three improvement strategies are effective and the convergence accuracy,convergence speed and robustness of the improved algorithm are significantly enhanced.

Keywords Equilibrium optimizer,Bidirectional chaotic opposition learning,Algorithm fusion,Random difference,Swarm intelligence optimization algorithms

1 引言

在过去几十年里,不同领域的优化问题变得更加复杂和困难,传统的数值方法很难解决这些问题。智能优化算法由于结构简单,可以在不计算目标函数导数的情况下找到问题最优解,在解决复杂的优化问题方面具有一定优势。智能优化算法包括松鼠搜索算法[1]、哈里斯鹰优化算法[2]和旗鱼优化算法[3]等,这些算法被研究者应用到很多领域[4-6]。

平衡优化器算法(Equilibrium Optimizer,EO)[7]是由Faramarzi等在2020年提出的一种新的智能优化算法,其结构简单,易于实现,求解能力强,因此被广泛应用于0-1背包问题[8]、特征选择[9]、股票市场预测[10]、最优路由维护[11]、冠状病毒疾病图像分割[12]等多个领域。没有免费午餐定理[13]告诉我们,没有一种方法能够解决所有的优化问题,这促使

到稿日期:2022-11-16 返修日期:2023-02-07
基金项目:国家自然科学基金(61772013);江苏省青年基金(BK20190578)
This work was supported by the National Natural Science Foundation of China(61772013) and Youth Foundation of Jiangsu(BK20190578).
通信作者:鲁海燕(luhaiyan@jiangnan.edu.cn)

研究人员在开发新算法的同时改进原始算法。EO 仅仅依靠物理原理得到，其全局探索能力弱于局部勘探能力，求解约束优化问题时可能会出现早熟或局部停滞。为了提高 EO 的寻优性能，一些学者对其进行了改进。Hemalatha 等[14]利用高斯变异和基于种群划分的搜索机制来避免算法收敛到次优解，提高了算法收敛速度。Ouadfel 等[15]提出一种基于 Relief 过滤器算法和 EO 的混合算法来提高 EO 的局部开发能力。Jia 等[16]将 EO 和热交换优化算法融合来提高 EO 的收敛精度。虽然以上方法都对 EO 有了改进，但是改进 EO 仍然存在收敛精度低、探索能力较弱、容易出现早熟收敛、在高维函数上结果不稳定等问题。

针对上述问题，本文提出一种融合麻雀搜索和随机差分的双向学习平衡优化器算法（Bidirectional Learning Equilibrium Optimizer Combining Sparrow Search and Random Difference，SRB-EO）。首先，利用麻雀搜索算法（Sparrow Search Algorithm，SSA）[17]来提出自适应种群划分策略，以平衡 EO 的探索和勘探能力，从而提高算法的收敛精度和速度。其次，结合随机差分策略改进平衡池，以增加粒子之间的信息交流，从而提高算法的全局探索能力。最后，提出双向混沌反向学习策略来优化更新后的种群，增加 EO 的种群多样性以避免收敛到局部最优，从而进一步提高算法的收敛精度。

2 研究基础

本章介绍平衡优化器算法和麻雀搜索算法的基本原理和内部结构。

2.1 平衡优化器算法

平衡优化器算法的灵感来自于控制体积上简单的动态质量平衡，物理原理为控制体积质量随时间的变化与进出系统质量的差值相等。EO 运用随机初始化方式构造种群，初始化公式如下：

$$\vec{C}_i=\vec{C}_{\min}+\boldsymbol{rand}\times(\vec{C}_{\max}-\vec{C}_{\min}),i=1,2,\cdots,N \tag{1}$$

其中，$\vec{C}_i$ 表示第 i 个粒子的初始化浓度向量，$\boldsymbol{rand}$ 是分量位于(0,1)上的 D 维随机向量，$\vec{C}_{\max}$ 和 $\vec{C}_{\min}$ 是 $\vec{C}_i$ 的上界和下界向量。在算法迭代中，用如下公式更新浓度向量：

$$\vec{C}_i^{ite+1}=\vec{C}_{eq}+(\vec{C}_i^{ite}-\vec{C}_{eq})\cdot\vec{F}+\frac{\vec{G}}{\vec{\lambda}V}(1-\vec{F}) \tag{2}$$

其中，$\vec{C}_i^{ite}$ 是第 ite 次迭代的第 i 个浓度向量，$\vec{\lambda}$ 是元素介于[0,1]的向量，V 是单位体积，$\vec{C}_{\text{eq}}$ 是从平衡池中随机选择的向量。平衡池构造如下：

$$\vec{C}_{\text{eq,pool}}=\{\vec{C}_{\text{eq}(1)},\vec{C}_{\text{eq}(2)},\vec{C}_{\text{eq}(3)},\vec{C}_{\text{eq}(4)},\vec{C}_{\text{ave}}\} \tag{3}$$

其中，向量 $\vec{C}_{\text{eq}(1)},\vec{C}_{\text{eq}(2)},\vec{C}_{\text{eq}(3)},\vec{C}_{\text{eq}(4)},\vec{C}_{\text{ave}}$ 分别是当前迭代中 4 个最好的浓度向量和它们的平均向量。向量 $\vec{F}$ 的表达式由下式给出：

$$\vec{F}=\mathrm{e}^{\vec{\lambda}(t-t_0)} \tag{4}$$

其中，时间 t 的公式如下：

$$t=\left(1-\frac{ite}{T}\right)^{\left(a_2\frac{ite}{T}\right)} \tag{5}$$

其中，a_2 是控制常量，ite 和 T 分别为当前迭代次数和最大迭代次数；t_0 具体如下：

$$t_0=\frac{1}{\vec{\lambda}}\ln(-a_1\operatorname{sign}(\vec{r}-0.5)[1-\mathrm{e}^{-\vec{\lambda}t}])+t \tag{6}$$

其中，a_1 是控制常量，$\vec{r}$ 是元素介于[0,1]的向量。将式(6)中给出的表达式代入式(4)后，得到的 $\vec{F}$ 如下所示：

$$\vec{F}=a_1\operatorname{sign}(\vec{r}-0.5)[\mathrm{e}^{-\vec{\lambda}t}-1] \tag{7}$$

式(2)的生成率 $\vec{G}$ 是 EO 中的主要影响因素之一，该参数的值定义如下：

$$\vec{G}=G_0\,\mathrm{e}^{-\vec{\lambda}(t-t_0)} \tag{8}$$

$$\vec{G}_0=\overrightarrow{GCP}(\vec{C}_{\text{eq}}-\vec{\lambda}\vec{C}) \tag{9}$$

$$\overrightarrow{GCP}=\begin{cases}0.5r_1, & r_2\geqslant GP\\ 0, & r_2<GP\end{cases} \tag{10}$$

其中，GP 为生成率概率，$\overrightarrow{GCP}$ 为生成率 $\vec{G}$ 的控制参数，$\vec{G}_0$ 控制是否使用 $\overrightarrow{GCP}$ 来更新浓度向量，r_1 和 r_2 是[0,1]中均匀分布的随机数。

2.2 麻雀搜索算法

麻雀搜索算法的种群分为两部分，即发现者和跟随者，它们有 3 种行为：觅食、跟踪和侦察。其中，发现者位置的公式如下：

$$\boldsymbol{X}_{i,j}^{t+1}=\begin{cases}\boldsymbol{X}_{i,j}^{t}\cdot\exp\left(\dfrac{-i}{\alpha\cdot M}\right), & R_2<ST\\ \boldsymbol{X}_{i,j}^{t}+\boldsymbol{Q}\cdot\boldsymbol{L}, & R_2\geqslant ST\end{cases} \tag{11}$$

其中，t 和 M 分别表示当前迭代次数和最大迭代次数，$\boldsymbol{X}_{i,j}^{t}$ 为第 i 个麻雀的位置向量，j 代表维数，α 和 R_2 都表示介于[0,1]的随机数，R_2 是飞行参数，ST 是安全阈值参数，$\boldsymbol{L}$ 是所有元素为 1 的 $1\times D$ 矩阵，Q 是服从正态分布的随机数。跟随者的位置更新描述如下：

$$\boldsymbol{X}_{i,j}^{t+1}=\begin{cases}Q\cdot\exp\left(\dfrac{\boldsymbol{X}_{\text{worst}}^{t}-\boldsymbol{X}_{i,j}^{t}}{i^2}\right), & i>\dfrac{n}{2}\\ \boldsymbol{X}_{\text{best}}^{t+1}+|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{\text{best}}^{t+1}|\cdot\boldsymbol{A}^{+}\cdot\boldsymbol{L}, & i\leqslant\dfrac{n}{2}\end{cases} \tag{12}$$

其中，X_{best}^{t+1} 为在 $t+1$ 次迭代中食物资源最好的位置，X_{worst}^{t} 为在第 t 次迭代中食物资源最差的位置，$\boldsymbol{A}$ 为元素随机赋值为 1 或 -1 的 $1\times D$ 矩阵，$\boldsymbol{A}^{+}=\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}$，$Q$ 是服从正态分布的随机数。当发现危险时，麻雀种群将采取警戒行动，该行为公式如下：

$$\boldsymbol{X}_{i,j}^{t+1}=\begin{cases}X_{\text{best}}^{t}+\beta\cdot|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{\text{best}}^{t}|, & f_i>f_g\\ \boldsymbol{X}_{i,j}^{t}+K\cdot\left(\dfrac{|\boldsymbol{X}_{i,j}^{t}-\boldsymbol{X}_{\text{worst}}^{t}|}{(f_i-f_w)+\varepsilon}\right), & f_i=f_g\end{cases} \tag{13}$$

其中，β 和 K 都是随机数，前者元素服从标准正态分布，后者元素介于[-1,1]，ε 为常数，f_i 表示第 i 个体的适应度值，f_g 和 f_w 分别是当前迭代中的最优与最差适应度值。

3 多策略改进平衡优化器算法

3.1 融合麻雀搜索算法策略

EO 具有设置参数少、求解能力强、易于实现等优点，但

是该算法仅基于物理原理，结构单一，因此存在收敛速度慢、收敛精度低、容易陷入局部停滞等缺陷。SSA 有着收敛速度快、收敛精度高的特点，因此本文结合 SSA 的种群划分策略和结构特点，对 EO 进行改进。

3.1.1 自适应种群划分策略

SSA 根据适应度值对种群个体进行排序，适应度较好的部分个体为发现者，其余个体为跟随者。这种种群划分策略使得算法收敛速度更快，搜索能力更强，因此本文依据这种策略对 EO 的种群进行划分，以提高 EO 的收敛速度和精度。

SSA 中发现者对跟随者具有引导作用，因此发现者的比例随迭代次数的增加而增加可以提高算法的寻优性能，并更好地平衡算法的全局探索能力和局部勘探能力。本文提出自适应种群划分策略来对 EO 的种群进行分类。改进 EO 中发现者的比例系数设置如下：

$$PN=PN_{\min}+(PN_{\max}-PN_{\min})*\left(\frac{ite}{T}\right)^2 \tag{14}$$

其中，PN 为发现者的比例，$PN_{\min}$ 和 $PN_{\max}$ 为发现者在种群中占比的最小值和最大值。发现者的个数如下：

$$N_{PN}=\text{round}(PN*N) \tag{15}$$

其中，N 为种群数量，round 为取整函数。

在迭代前期，改进 EO 的种群大范围地探索最优值，发现者比例较小有利于提高算法的全局搜索能力。在迭代后期，种群中更多的个体能够搜索到全局最优值，而发现者又代表最优的部分个体，因此提高发现者比例有利于增强算法的局部勘探能力。

3.1.2 迭代公式更新

首先，SSA 中发现者的任务是找到优质食物并引导跟随者前来觅食。本文结合 SSA 发现者位置更新公式中的自适应指数来改进平衡池中的粒子，然后对 EO 的浓度更新公式进行改进，并将其作为改进 EO 中发现者的位置更新公式，如下所示：

$$\vec{C}_i^{\,ite+1}=\vec{C}_{\text{neq}}+(\vec{C}_i^{\,ite}-\vec{C}_{\text{neq}})\cdot\vec{F}+\frac{\vec{G}}{\vec{\lambda}V}(1-\vec{F}) \tag{16}$$

$$\vec{C}_{\text{neq}}=\exp\left(-\frac{ite}{T}\right)\cdot\vec{C}_{eq} \tag{17}$$

其中，ite 和 T 分别为当前迭代次数和最大迭代次数，$\vec{C}_i^{\,ite}$ 为第 ite 次中第 i 个粒子的位置向量，$\vec{\lambda}$ 为元素介于[0,1]的向量，V 为单位体积，$\vec{C}_{eq}$ 由平衡池随机选取。

其次，SSA 中没有找到食物的跟随者需要去别处觅食，找到食物的跟随者将继续紧跟发现者来寻找食物。利用 SSA 的这种启发，改进 EO 中的跟随者进行如下位置更新：

$$\vec{C}_i^{\,ite+1}=\begin{cases}Q\cdot\exp\left(\dfrac{\vec{C}_{\text{worst}}^{\,ite}-\vec{C}_i^{\,ite}}{i^2}\right), & i>\dfrac{n}{2}\\ \vec{C}_{\text{eq}(1)}^{\,ite+1}+|\vec{C}_i^{\,ite}-\vec{C}_{\text{eq}(1)}^{\,ite+1}|\cdot\boldsymbol{A}^{+}\cdot\boldsymbol{L}, & i\leqslant\dfrac{n}{2}\end{cases} \tag{18}$$

其中，$\vec{C}_{\text{eq}(1)}^{\,ite+1}$ 为发现者在 $ite+1$ 次迭代中的最佳位置，$\vec{C}_{\text{worst}}^{\,ite}$ 为种群在 ite 次迭代中最差个体的位置，$\boldsymbol{A}$ 是元素随机赋值为 1 或 −1 的 $1\times D$ 矩阵，$\boldsymbol{A}^{+}=\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}$，$Q$ 是服从正态分布的随机数。

3.2 随机差分重构平衡池策略

平衡池是 EO 特有的结构，虽然它能利用一些信息，但是只有当前粒子和精英粒子之间的信息共享，因此算法寻优的引导是单一的，且不利于提高 EO 中的信息利用效率。

差分进化算法(Differential Evolution Algorithm，DE)[18]通过个体间的差分量实现变异，是指导“适者生存”进化原则的一种高效、有效和稳健的优化方法，在连续变量的问题中有很大的优势。因此，本文参考差分进化的思想，提出随机差分重构平衡池策略，其原理为随机挑选的两个浓度向量通过差分变异来产生成新向量，以构建新的平衡池。具体公式如下：

$$\vec{C}_{\text{eq}(5)}=rand*(\vec{C}_a-\vec{C}_b) \tag{19}$$

$$\vec{C}_{\text{eq}(6)}=rand*(\vec{C}_c-\vec{C}_d) \tag{20}$$

$$\vec{C}_{\text{ave}}=\left(\sum_{i=1}^{6}\vec{C}_{\text{eq}(i)}\right)/6 \tag{21}$$

$$\vec{C}_{\text{pool}}=\{\vec{C}_{\text{eq}(1)},\vec{C}_{\text{eq}(2)},\cdots,\vec{C}_{\text{eq}(6)},\vec{C}_{\text{ave}}\} \tag{22}$$

其中，$\vec{C}_a,\vec{C}_b,\vec{C}_c,\vec{C}_d$ 为随机挑选的个体，$\vec{C}_{\text{pool}}$ 为新的平衡池。在迭代初期，改进算法种群的个体差异较大，增强了全局搜索能力。在迭代后期，改进算法的种群个体中差异较小，可以减小搜索范围，提高局部开发能力。总体来看，随机差分重构平衡池策略加强了个体间信息的交流，有利于算法全局寻优。

3.3 双向混沌反向学习策略

为了进一步改善 EO 探索弱于开发和容易早熟收敛的问题，本文结合精英反向学习策略[19]的思想，提出双向混沌反向学习策略。该策略的原理是在迭代前期先对个体进行反向学习，在迭代后期对个体进行混沌变异，最后利用贪婪策略选取适应度值较优的个体来组成进入下一次迭代的新种群。

当迭代次数 $ite<T/2$ 时，改进算法对浓度更新后的种群采用反向学习策略。反向点的定义如下：

设 $P(x_1,x_2,\cdots,x_n)$ 为 n 维坐标系的点，$x_1,\cdots,x_n\in\Psi$，$x_i\in[a_i,b_i]$，反向点 P^{OB} 的坐标为 $x_1^{\text{OB}},\cdots,x_n^{\text{OB}}$，其中：

$$x_i^{\text{OB}}=a_i+b_i-x_i;i=1,2,\cdots,n \tag{23}$$

由此得到新的浓度更新公式为：

$$\vec{C}_i^{\text{new}}=\boldsymbol{UB}+\boldsymbol{LB}-\vec{C}_i \tag{24}$$

其中，$\boldsymbol{UB}$ 和 $\boldsymbol{LB}$ 分别为种群的上界向量与下界向量。

当迭代次数 $ite\geqslant T/2$ 时，改进算法用混沌映射产生的序列对浓度更新后的种群进行变异。混沌映射[20]是一种由确定性系统产生的随机性序列，常见的混沌映射包括 Piecewise 映射、Sine 映射。为了直观展示以上两种混沌映射的效果，图 1 和图 2 给出了两种映射的时序图。由时序图可以看出 Sine 混沌序列的随机性比 Piecewise 混沌序列更强，因此选用 Sine 映射对更新后的个体进行扰动，公式如下：

$$\vec{C}_i^{\text{new}}=s_i\times\vec{C}_i \tag{25}$$

其中，s_i 表示一个 D 维 Sine 混沌序列($d=1,2,\cdots,D$)，其混沌映射公式如下：

$$s_i^{d+1}=\frac{a}{4}\times\sin(\pi s_i^d) \tag{26}$$

其中，s_i^d 为第 i 个混沌序列中的第 d 个位置的随机数，$a=4$。

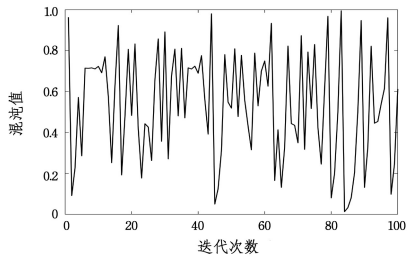


图 1 Piecewise 混沌映射的时序图

Fig. 1 Time series diagram of Piecewise chaotic mapping

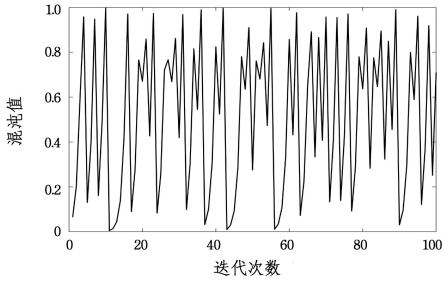


图 2 Sine 混沌映射的时序图

Fig. 2 Time series diagram of Sine chaotic mapping

3.4 多策略融合后的 SRB-EO 算法

SRB-EO 首先融合 SSA 的种群划分策略，采用自适应种群划分策略将种群分为发现者和跟随者，两者分别依式(16)和式(18)更新；其次，用随机差分策略生成两个新解，并将其加入平衡池中；最后，利用双向混沌反向学习对更新后的个体进行扰动，再贪婪选择适应度较好的个体进入下一次迭代。SRB-EO 算法的伪代码如算法 1 所示。

算法 1 SRB-EO 算法

输入：种群大小 N，维数 d，最大迭代次数 T，上界向量$\vec{C}_{max}$和下界向量$\vec{C}_{min}$，单位体积 V，控制常量 a_1 和 a_2，生成率概率 GP，发现者的最大比例 PN_{max}和最小比例 PD_{min}

输出：$\vec{C}_{eq(1)}$及其最优适应度值 f_g

1. 由式(1)随机初始化种群//初始化
2. 计算适应度值
3. 根据适应度值对种群进行排序，找到$\vec{C}_{eq(1)}$至$\vec{C}_{eq(4)}$，$\vec{C}_{worst}$
4. 利用随机差分策略生成$\vec{C}_{eq(5)}$和$\vec{C}_{eq(6)}$//构建新的平衡池
5. for ite=1 to T do
6. 更新$\vec{C}_{eq(1)}$至$\vec{C}_{eq(6)}$
7. 计算$\vec{C}_{ave}$及构造平衡池
8. for i=1 to PD * N do
9. 在平衡池中找到$\vec{C}_{eq}$
10. 由式(16)来更新种群
11. end for//更新发现者种群
12. for i=PD * N+1 to N do
13. 由式(18)更新跟随者
14. end for//更新跟随者种群
15. if ite<T/2
16. for i=1 to N do
17. 由式(24)生成反向种群
18. end for
19. else if
20. 由式(25)对种群进行混沌变异
21. end if//双向混沌反向学习策略
22. for i=1 to N do
23. 根据适应度值贪婪选择个体
24. end for
25. 对种群进行越界处理
26. end for//算法结束

4 SRB-EO 算法时间复杂度

种群规模 N、个体维数 D、最大迭代次数 T 和适应度函数的运算耗时 F，都是影响算法时间复杂度的重要指标。因为 EO 的初始化阶段、适应度函数计算和迭代过程的时间复杂度分别为 $O_1=O(D\cdot N)$，$O_2=O(T\cdot F\cdot N)$和 $O_3=O(T\cdot D\cdot N)$，所以 EO 的时间复杂度为：

$$\begin{aligned}O(EO)&=O_1+O_2+O_3\\&=O(D\cdot N)+O(T\cdot F\cdot N)+O(T\cdot D\cdot N)\\&=O(T\cdot F\cdot N+T\cdot D\cdot N)\\&=O(TN(F+D))\end{aligned}$$

SRB-EO 与 EO 相比，初始化阶段时间复杂度不变；SRB-EO 将种群分成发现者和跟随者两部分：发现者的时间复杂度 $O_4=O(T\cdot PD\cdot N\cdot D)$，跟随者的时间复杂度 $O_5=O(T\cdot(N-PD\cdot N)\cdot D)$；SRB-EO 用随机差分来重构平衡池的时间复杂度 $O_6=O(2\cdot T\cdot D)$；SRB-EO 引入双向混沌反向学习策略后，当 $ite<T/2$ 时，SRB-EO 利用反向学习更新种群，其时间复杂度 $O_7=O(0.5\cdot T\cdot N\cdot D)$，当 $ite\geq T/2$ 时，SRB-EO 对更新后的种群进行 Sine 混沌变异，其时间复杂度 $O_8=O(0.5\cdot T\cdot N\cdot D)$；最后结合贪婪策略选择个体的时间复杂度 $O_9=O(T\times F\times N)$。通过以上分析，SRB-EO 的时间复杂度如下：

$$\begin{aligned}O(SRB\text{-}EO)&=O_1+O_2+O_4+O_5+O_6+O_7+O_8+O_9\\&=O(D\cdot N)+O(T\cdot F\cdot N)+O(T\cdot PD\cdot N\cdot D)+O(T\cdot(N-PD\cdot N)\cdot D)+O(2\cdot T\cdot D)+O(0.5\cdot T\cdot N\cdot D)+O(0.5\cdot T\cdot F\cdot N)+O(T\times F\times N)\\&=O(D\cdot N+2T\cdot F\cdot N+2T\cdot N\cdot D+2\cdot T\cdot D)\\&=O(T\cdot F\cdot N+T\cdot D\cdot N)\\&=O(TN(F+D))\end{aligned}$$

SRB-EO 在大幅提升 EO 整体性能的基础上，虽然增加了一部分计算量，但是总体未改变算法的时间复杂度。

5 实验结果与分析

5.1 实验环境

实验中计算机的操作系统为 Windows11，CPU 为 AMD Ryzen 7 5800H，主频 3.20GHz，内存 16GB，编程软件为 MATLAB 2022a。

5.2 算法与参数设置

为了验证 SRB-EO 的有效性，本文选取一些最新的智能优化算法和改进 EO 进行对比。实验中用到平衡优化器算法(EO)、麻雀搜索算法(SSA)、灰狼算法(Grey Wolf Optimizer, GWO)[21]、鲸鱼优化算法(Whale Optimization Algorithm, WOA)[22]、蜜獾算法(Honey Badger Algorithm, HBA)[23]、联合平衡优化器算法(UEO)[24]、基于 Lévy 飞行和迭代余弦算子的新型平衡优化器算法(MEO)[25]、基于 Tent 混沌和透镜成像学习策略的平衡优化器算法(TLIL-EO)[26]，以上算法的参数设置如表 1 所列。

表 1 算法参数设置

Table 1 Parameter setting of each algorithm

算法	参数设置
GWO	$a=2-t\times 2/T$
WOA	$a=2-t\times 2/T$
HBA	$\beta=6;C=2$
SSA	$PD=20\%;ST=80\%;SD=0.1$
EO	$V=1;a_1=2;a_2=1;GP=0.5$
UEO	$V=1;a_1=2;a_2=1;GP=0.5$
MEO	$V=1;a_1=2;a_2=1;U_p=0.5$
TLIL-EO	$V=1;a_1=2;a_2=1;GP=0.5;k=2$
SRB-EO	$PN_{min}=0.2;PN_{max}=0.4;V=1;a_1=2;a_2=1;GP=0.5$

5.3 测试函数

本文通过 14 个具有代表性的测试函数[27]，对改进算法 SRB-EO 的优化性能进行评价。14 个测试函数分为：单峰函数(F_1-F_6)、多峰函数(F_7-F_{11})和固定维多峰函数($F_{12}-F_{14}$)。以上测试函数的信息如表 2 所列。

表 2 测试函数

Table 2 Test functions

函数	函数名	维数	搜索范围	最优值	特征
F_1	Sphere	30	[−100,100]	0	单峰
F_2	Schwefel 2.22	30	[−10,10]	0	单峰
F_3	Schwefel 1.2	30	[−100,100]	0	单峰
F_4	Schwefel 2.21	30	[−100,100]	0	单峰
F_5	Rosenbrock	30	[−30,30]	0	单峰
F_6	Step	30	[−100,100]	0	单峰
F_7	Rastrigin	30	[−5.12,5.12]	0	多峰
F_8	Ackley	30	[−32,32]	0	多峰
F_9	Griewank	30	[−600,600]	0	多峰
F_{10}	Penalized1	30	[−50,50]	0	多峰
F_{11}	Penalized2	30	[−50,50]	0	多峰
F_{12}	Kowalik	4	[−5,5]	0.0003	固定维
F_{13}	Hartman2	6	[0,1]	−3.32	固定维
F_{14}	Shekel5	4	[0,10]	−10.1532	固定维

5.4 参数敏感性分析

SRB-EO 包括融合麻雀搜索与自适应种群划分策略、随机差分重构平衡池策略和双向混沌反向学习策略 3 种策略。在融合麻雀搜索与自适应种群划分策略的改进 EO(ASSA-EO)中，发现者最大比例 PN_{min} 和最小比例 PN_{min} 的取值不仅会影响 ASSA-EO 的寻优性能，而且会影响 SRB-EO 的寻优能力。为了选择合适的参数值，本节选取 14 个测试函数中 EO 容易陷入局部最优的 PN_{min} 进行实验。表 3 列举了 ASSA-EO 的 6 种参数取值情况，为保证实验的公平性，算法的种群规模设为 30，迭代次数设为 500。表 4 所列结果为不同算法独立运行 30 次的平均值。

表 3 ASSA-EO 的不同参数设置

Table 3 Different parameters setting of ASSA-EO

改进算法	PN_{min}	PN_{min}
ASSA-EO1	0.2	0.4
ASSA-EO2	0.2	0.6
ASSA-EO3	0.2	0.8
ASSA-EO4	0.4	0.6
ASSA-EO5	0.4	0.8
ASSA-EO6	0.6	0.8

表 4 不同参数设置下 ASSA-EO 的实验结果

Table 4 Experimental results of ASSA-EO with different parameters setting

算法	PN_{min}	PN_{min}	PN_{min}	PN_{min}
EO	9.45×10^{-23}	1.00×10^{-9}	2.54×10^{1}	2.40×10^{-2}
ASSA-EO1	$\mathbf{1.28\times10^{-119}}$	$\mathbf{3.34\times10^{-194}}$	$\mathbf{1.58\times10^{-5}}$	$\mathbf{2.78\times10^{-8}}$
ASSA-EO2	4.20×10^{-118}	3.32×10^{-192}	2.00×10^{1}	1.50×10^{-1}
ASSA-EO3	1.16×10^{-118}	5.98×10^{-192}	2.89×10^{1}	2.98
ASSA-EO4	1.97×10^{-118}	1.65×10^{-186}	2.10×10^{1}	2.87×10^{-1}
ASSA-EO5	9.15×10^{-118}	7.80×10^{-187}	2.30×10^{1}	5.07×10^{-1}
ASSA-EO6	8.71×10^{-116}	2.52×10^{-183}	2.47×10^{1}	8.27×10^{-1}

由表 4 可知，ASSA-EO1 在 PN_{min} 上跳出局部最优的能力更强，而且在其他测试函数上得到的结果最优。当参数 PN_{min} PN_{max} 分别取 0.2 和 0.4 时，ASSA-EO 可以在迭代前期进行大范围的探索，并且在迭代后期也不会因为跟随者比例过高而影响发现者的引导能力，因此 ASSA-EO 与 SRB-EO 采用该种参数设置。

5.5 改进策略有效性分析

为了验证 SRB-EO 中 3 种改进策略的有效性，将 SRB-EO 与 EO、SSA、融合麻雀搜索策略的 EO(SSA-EO)、融合麻雀搜索与自适应种群划分策略的 EO(ASSA-EO)、引入随机差分重构平衡池策略的 EO(RD-EO)、结合双向混沌反向学习策略的 EO(CO-EO)进行比较。为保证对比实验的公平性，本节实验设置种群规模为 30，迭代次数为 500，独立运行 30 次，记录实验结果，最优结果加粗标出，如表 5 所列。由表 5 可知，在 14 个测试函数上，SSA 较 EO 在整体收敛精度和鲁棒性上有一定的优势，但是仍然存在收敛精度低的问题。SSA-EO 在 F_1-F_4 上的收敛精度远远优于 EO 和 SSA，虽然在 F_5 上 EO 偏离理论最优值，但是 SSA-EO 却接近理论最优值，说明 SSA 的融合平衡了 EO 的全局探索和局部勘探，避免了陷入局部最优。在 F_6, F_{10}, F_{11} 上，SSA-EO 虽然比 EO 收敛精度略低几个数量级，但是相比 EO 还是有很大的精度优势。相较于原始算法 EO 和 SSA，SSA-EO 在其他测试函数上的收敛精度和鲁棒性都有所提高。ASSA-EO 在单峰函数上明显优于 SSA-EO，且在多峰和固定维函数上的精度和鲁棒性都有提高，因此自适应种群划分策略进一步平衡了融合算法的全局探索和局部勘探，优化效果明显。综合来看，ASSA-EO 解决了 EO 在多峰函数上陷入局部最优的问题，也能提高 SSA 的整体寻优精度，ASSA-EO 比两个原始算法更有优势。RD-EO 的整体寻优精度和鲁棒性相比 EO 均有所提高，说明差分进化重构平衡池策略加强了种群间信息的交流，有利于提高算法的全局寻优能力。CO-EO 在测试函数上的平均值和标准差均优于 EO，这是由于混沌映射和反向学习策略丰富了种群多样性，从而增强了算法跳出局部最优的能力，而且优质个体有利于引导其他个体寻优，贪婪策略可以让更优个体保留下来，以提高算法的收敛精度。

表 5 不同改进策略的对比实验结果

Table 5 Comparative experimental results of different improved strategies

函数	指标	EO	SSA	SSA-EO	ASSA-EO	RD-EO	CO-EO	SRB-EO
F_1	Mean	1.00×10^{-40}	9.38×10^{-58}	9.53×10^{-171}	1.34×10^{-225}	8.85×10^{-128}	**0.00**	**0.00**
	Std	3.12×10^{-40}	3.64×10^{-57}	0.00	0.00	4.14×10^{-127}	**0.00**	**0.00**
F_2	Mean	9.45×10^{-23}	9.92×10^{-29}	3.19×10^{-87}	1.28×10^{-119}	1.35×10^{-65}	8.94×10^{-201}	$\mathbf{1.50\times10^{-202}}$
	Std	1.59×10^{-22}	5.43×10^{-28}	9.28×10^{-87}	$2.93\times10^{-11}6$	3.14×10^{-65}	0.00	**0.00**
F_3	Mean	1.10×10^{-8}	2.81×10^{-23}	1.64×10^{-126}	3.34×10^{-194}	9.67×10^{-83}	**0.00**	**0.00**
	Std	5.43×10^{-8}	1.54×10^{-22}	$6.05\times10^{-12}6$	0.00	4.49×10^{-82}	**0.00**	**0.00**
F_4	Mean	4.75×10^{-10}	1.14×10^{-26}	1.10×10^{-80}	1.83×10^{-109}	2.78×10^{-58}	1.62×10^{-196}	$\mathbf{5.42\times10^{-201}}$
	Std	1.37×10^{-9}	6.20×10^{-26}	4.24×10^{-80}	5.74×10^{-102}	8.75×10^{-58}	0.00	**0.00**
F_5	Mean	2.54×10^{1}	5.91×10^{-5}	9.12×10^{-5}	1.58×10^{-5}	2.58×10^{1}	2.53×10^{1}	$\mathbf{1.25\times10^{-6}}$
	Std	1.96×10^{-1}	1.58×10^{-4}	2.03×10^{-5}	6.84×10^{-5}	1.79×10^{-1}	2.12×10^{-1}	$\mathbf{2.61\times10^{-5}}$
F_6	Mean	1.13×10^{-5}	$\mathbf{1.80\times10^{-11}}$	5.92×10^{-7}	5.87×10^{-8}	5.07×10^{-4}	3.07×10^{-5}	3.67×10^{-8}
	Std	6.49×10^{-6}	$\mathbf{4.69\times10^{-11}}$	2.96×10^{-6}	1.65×10^{-7}	2.18×10^{-4}	4.43×10^{-5}	1.07×10^{-7}
F_7	Mean	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Std	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F_8	Mean	8.23×10^{-15}	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$
	Std	1.30×10^{-15}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F_9	Mean	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Std	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F_{10}	Mean	6.91×10^{-3}	$\mathbf{2.43\times10^{-12}}$	7.83×10^{-9}	5.31×10^{-9}	5.23×10^{-5}	1.28×10^{-6}	1.41×10^{-9}
	Std	2.63×10^{-2}	$\mathbf{7.49\times10^{-12}}$	1.67×10^{-8}	1.48×10^{-8}	3.34×10^{-5}	1.63×10^{-6}	3.39×10^{-9}
F_{11}	Mean	2.40×10^{-2}	$\mathbf{2.19\times10^{-11}}$	2.12×10^{-8}	2.78×10^{-8}	4.30×10^{-2}	1.65	4.34×10^{-8}
	Std	3.86×10^{-2}	$\mathbf{7.35\times10^{-11}}$	4.51×10^{-8}	8.42×10^{-8}	5.69×10^{-2}	3.68×10^{-1}	1.01×10^{-8}
F_{12}	Mean	3.69×10^{-3}	3.16×10^{-4}	1.33×10^{-3}	4.10×10^{-4}	3.49×10^{-4}	3.38×10^{-4}	$\mathbf{3.01\times10^{-4}}$
	Std	7.59×10^{-3}	7.93×10^{-5}	1.46×10^{-3}	4.97×10^{-4}	9.07×10^{-5}	1.67×10^{-4}	$\mathbf{3.85\times10^{-6}}$
F_{13}	Mean	−3.27	−3.29	−3.21	−3.21	−3.26	−3.28	−3.32
	Std	5.92×10^{-2}	5.11×10^{-2}	-3.19×10^{-1}	7.71×10^{-2}	8.00×10^{-2}	6.15×10^{-2}	$\mathbf{7.70\times10^{-4}}$
F_{14}	Mean	−8.29	−7.77	-1.02×10^{1}	-1.02×10^{1}	−5.06	−8.61	$\mathbf{-1.02\times10^{1}}$
	Std	2.50	2.59	-1.32×10^{-2}	2.16×10^{-3}	8.88×10^{-16}	2.59×10^{-3}	$\mathbf{5.34\times10^{-4}}$

注："Mean"表示平均值，"Std"表示标准差。

综上所述，SSA-EO，RD-EO 和 CO-EO 的实验结果整体优于 EO，说明 SRB-EO 的 3 种改进策略都能提高算法性能。SRB-EO 比 SSA-EO，RD-EO 和 CO-EO 的平均值更接近理论最优值，且标准差更小，验证了 3 种策略相加比单一策略效果好，能够更好地提高 EO 的寻优精度和鲁棒性。

5.6 SRB-EO 与其他智能优化算法的对比分析

为了进一步说明 SRB-EO 的优势，选取表 1 中的算法和表 2 中的测试函数对 SRB-EO 进行实验且参数设置不变。表 6 列出了实验数据，并加粗标出表中的最优结果。由表 6 可知，SRB-EO 在实验中达到的最优值都比其他算法更接近理论最优值，说明 SRB-EO 的寻优能力和收敛精度更强。在 F_5 上，几乎所有的算法都陷入局部最优，但是 SRB-EO 的实验结果与理论最优值相差不大，说明其算法可以跳出局部最优。在 F_7 和 F_9 上，EO 与 4 种改进 EO 效果相当，平均值都能达到理论最优值，说明改进策略保持了 EO 原有的一些优良性能。在 F_{13} 和 F_{14} 上，虽然几乎所有算法的最优值都可以达到理论最优值，但是 SRB-EO 的平均值更接近理论最优值，标准差也比其他 3 种改进 EO 更小，说明 SRB-EO 的稳定性和鲁棒性更优。

表 6 不同算法的寻优结果对比

Table 6 Comparison of optimization results of different algorithms

函数	指标	GWO	WOA	HBA	EO	UEO	MEO	TLIL-EO	SRB-EO
F_1	最优值	5.55×10^{-29}	7.81×10^{-90}	1.32×10^{-142}	9.98×10^{-44}	1.12×10^{-127}	3.14×10^{-228}	7.26×10^{-298}	**0.00**
	平均值	1.24×10^{-27}	7.53×10^{-72}	$7.36\times10^{-13}5$	1.68×10^{-40}	2.39×10^{-117}	3.06×10^{-218}	1.33×10^{-297}	**0.00**
	标准差	1.73×10^{-27}	4.12×10^{-71}	2.75×10^{-134}	7.75×10^{-40}	8.42×10^{-117}	**0.00**	**0.00**	**0.00**
F_2	最优值	2.23×10^{-16}	4.08×10^{-58}	5.32×10^{-75}	1.15×10^{-23}	1.32×10^{-67}	2.03×10^{-118}	1.09×10^{-148}	$\mathbf{8.29\times10^{-221}}$
	平均值	1.27×10^{-15}	2.56×10^{-50}	6.80×10^{-72}	6.58×10^{-23}	3.59×10^{-63}	1.45×10^{-114}	1.65×10^{-148}	$\mathbf{1.50\times10^{-202}}$
	标准差	8.80×10^{-16}	6.34×10^{-50}	1.27×10^{-71}	5.43×10^{-23}	1.93×10^{-62}	7.43×10^{-114}	5.09×10^{-149}	**0.00**
F_3	最优值	2.40×10^{-8}	2.34×10^{4}	3.40×10^{-107}	2.58×10^{-12}	3.19×10^{-72}	8.91×10^{-202}	$1.20\times10^{-29}7$	**0.00**
	平均值	2.01×10^{-5}	5.01×10^{4}	7.04×10^{-97}	4.99×10^{-9}	1.02×10^{-52}	1.41×10^{-193}	$3.99\times10^{-29}7$	**0.00**
	标准差	3.13×10^{-5}	1.51×10^{4}	3.69×10^{-96}	1.53×10^{-8}	5.01×10^{-52}	**0.00**	**0.00**	**0.00**
F_4	最优值	2.77×10^{-8}	2.74×10^{-1}	5.91×10^{-61}	2.28×10^{-12}	3.03×10^{-51}	5.90×10^{-109}	1.10×10^{-149}	$\mathbf{5.07\times10^{-208}}$
	平均值	7.33×10^{-7}	4.55×10^{1}	3.46×10^{-57}	2.41×10^{-10}	1.72×10^{-46}	1.10×10^{-103}	1.71×10^{-149}	$\mathbf{5.42\times10^{-201}}$
	标准差	7.79×10^{-7}	3.14×10^{1}	1.15×10^{-56}	3.11×10^{-10}	4.57×10^{-46}	3.84×10^{-103}	$1.97\times10^{-15}0$	**0.00**
F_5	最优值	2.60×10^{1}	2.72×10^{1}	2.24×10^{1}	2.50×10^{1}	2.44×10^{1}	2.79×10^{1}	2.48×10^{1}	$\mathbf{3.11\times10^{-10}}$
	平均值	2.71×10^{1}	2.79×10^{1}	2.41×10^{1}	2.54×10^{1}	2.52×10^{1}	2.82×10^{1}	2.52×10^{1}	$\mathbf{1.25\times10^{-6}}$
	标准差	7.44×10^{-1}	3.67×10^{-1}	9.02×10^{-1}	2.24×10^{-1}	4.02×10^{-1}	1.39×10^{-1}	1.86×10^{-1}	$\mathbf{2.61\times10^{-5}}$
F_6	最优值	2.50×10^{-1}	9.19×10^{-2}	4.87×10^{-6}	2.91×10^{-6}	1.24×10^{-5}	3.46	1.02×10^{-6}	$\mathbf{5.02\times10^{-13}}$
	平均值	7.74×10^{-1}	4.21×10^{-1}	1.86×10^{-2}	1.09×10^{-5}	9.15×10^{-2}	4.04	6.35×10^{-6}	$\mathbf{3.67\times10^{-8}}$
	标准差	4.01×10^{-1}	3.06×10^{-1}	6.35×10^{-2}	8.22×10^{-6}	1.67×10^{-1}	3.23×10^{-1}	5.33×10^{-6}	$\mathbf{1.07\times10^{-7}}$
F_7	最优值	5.68×10^{-14}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	平均值	1.64	3.79×10^{-15}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	标准差	2.37	2.08×10^{-14}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F_8	最优值	7.55×10^{-14}	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	4.44×10^{-15}	$\mathbf{8.88\times10^{-16}}$	$\mathbf{8.88\times10^{-16}}$	4.44×10^{-15}	$\mathbf{8.88\times10^{-16}}$
	平均值	1.03×10^{-13}	3.49×10^{-15}	$\mathbf{8.88\times10^{-16}}$	8.11×10^{-15}	1.72×10^{-15}	$\mathbf{8.88\times10^{-16}}$	4.44×10^{-15}	$\mathbf{8.88\times10^{-16}}$
	标准差	1.58×10^{-14}	2.63×10^{-15}	**0.00**	1.47×10^{-15}	1.53×10^{-15}	**0.00**	**0.00**	**0.00**

（续表）

函数	指标	GWO	WOA	HBA	EO	UEO	MEO	TLIL-EO	SRB-EO
	最优值	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F_9	平均值	2.70×10^{-3}	9.30×10^{-3}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	标准差	7.93×10^{-3}	3.59×10^{-2}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	最优值	8.71×10^{-6}	4.33×10^{-3}	8.28×10^{-7}	3.00×10^{-8}	7.04×10^{-7}	3.09×10^{-1}	2.32×10^{-8}	$\mathbf{2.49\times10^{-14}}$
F_{10}	平均值	4.32×10^{-2}	2.26×10^{-2}	4.44×10^{-4}	3.46×10^{-3}	3.90×10^{-3}	4.51×10^{-1}	4.87×10^{-7}	$\mathbf{1.41\times10^{-9}}$
	标准差	2.43×10^{-2}	1.23×10^{-2}	2.39×10^{-3}	1.89×10^{-2}	1.90×10^{-2}	8.60×10^{-2}	6.63×10^{-7}	$\mathbf{3.39\times10^{-9}}$
	最优值	1.98×10^{-1}	2.15×10^{-1}	1.12×10^{-2}	8.00×10^{-7}	9.73×10^{-2}	2.19	1.28×10^{-6}	$\mathbf{1.86\times10^{-12}}$
F_{11}	平均值	5.95×10^{-1}	5.25×10^{-1}	3.49×10^{-1}	3.71×10^{-2}	6.02×10^{-1}	2.43	1.42×10^{-1}	$\mathbf{4.34\times10^{-8}}$
	标准差	2.05×10^{-1}	2.38×10^{-1}	2.69×10^{-1}	5.67×10^{-2}	4.82×10^{-1}	1.15×10^{-1}	3.89×10^{-1}	$\mathbf{1.01\times10^{-8}}$
	最优值	3.08×10^{-4}	3.14×10^{-4}	3.07×10^{-4}	3.07×10^{-4}	1.83×10^{-2}	3.48×10^{-4}	3.07×10^{-4}	$\mathbf{3.00\times10^{-4}}$
F_{12}	平均值	4.42×10^{-3}	6.95×10^{-4}	6.33×10^{-3}	3.05×10^{-3}	2.66×10^{-1}	9.55×10^{-4}	3.16×10^{-4}	$\mathbf{3.01\times10^{-4}}$
	标准差	8.11×10^{-3}	5.44×10^{-4}	9.80×10^{-3}	6.91×10^{-3}	3.36×10^{-1}	1.16×10^{-3}	2.22×10^{-5}	$\mathbf{3.85\times10^{-6}}$
	最优值	**−3.32**	**−3.32**	**−3.32**	**−3.32**	**−3.32**	**−3.32**	**−3.32**	**−3.32**
F_{13}	平均值	−3.26	−3.21	−3.26	−3.25	−3.26	−3.21	−3.31	**−3.32**
	标准差	8.25×10^{-2}	1.89×10^{-1}	9.20×10^{-2}	7.24×10^{-2}	6.05×10^{-2}	1.08×10^{-1}	3.94×10^{-2}	$\mathbf{7.70\times10^{-4}}$
	最优值	$\mathbf{-1.02\times10^{1}}$	$\mathbf{-1.02\times10^{1}}$	$\mathbf{-1.02\times10^{1}}$	$\mathbf{-1.02\times10^{1}}$	$\mathbf{-1.02\times10^{1}}$	-1.00×10^{1}	$\mathbf{-1.02\times10^{1}}$	$\mathbf{-1.02\times10^{1}}$
F_{14}	平均值	−8.89	−7.77	$\mathbf{-1.02\times10^{1}}$	−8.64	−8.72	−6.14	−5.23	$\mathbf{-1.02\times10^{1}}$
	标准差	2.37	2.81	$\mathbf{5.60\times10^{-15}}$	2.62	2.70	2.51	9.31×10^{-1}	5.34×10^{-4}

为了直观地比较算法的收敛精度和速度，本文根据迭代次数和适应度值绘制收敛曲线图，如图 3 所示。

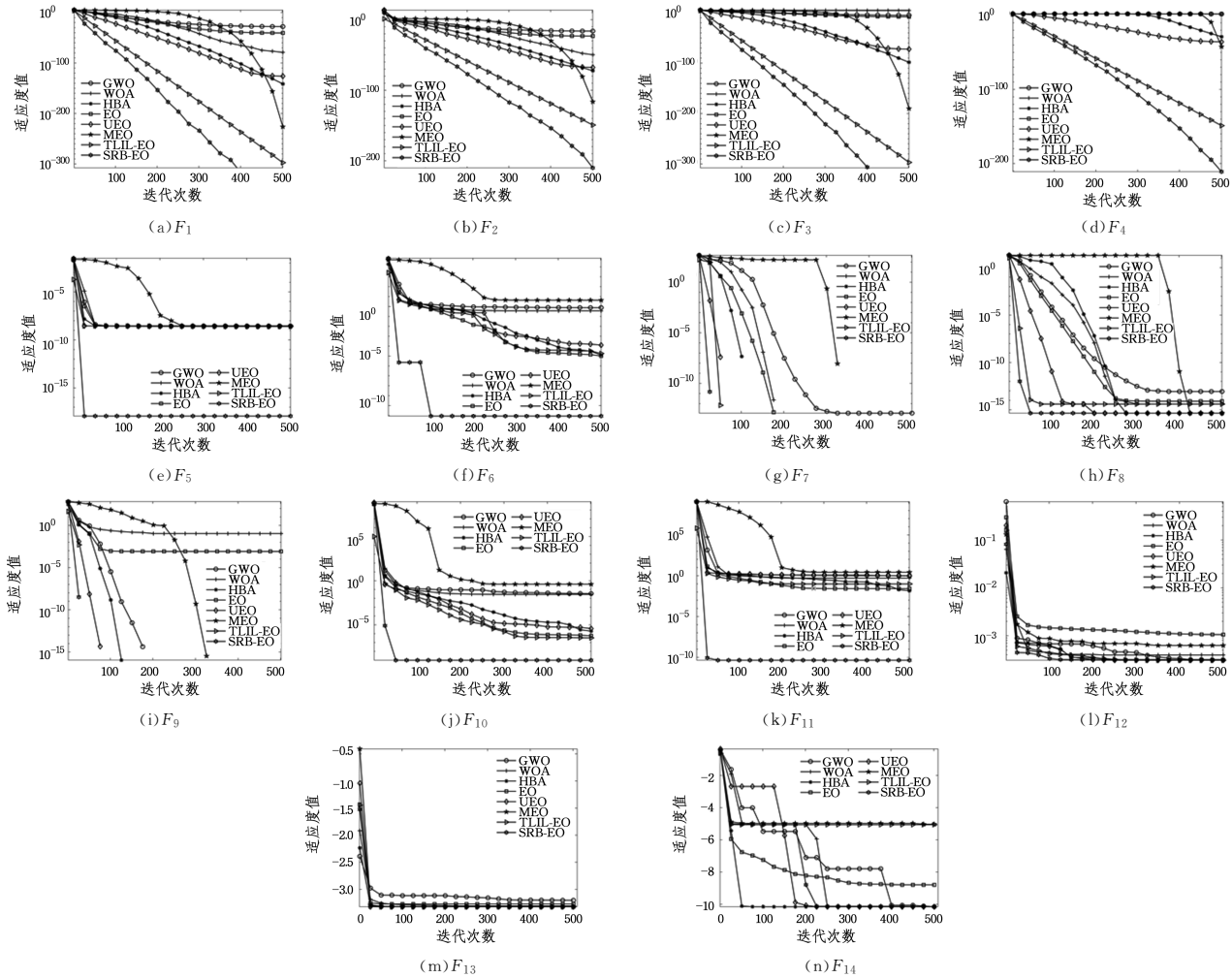


图 3 算法在测试函数上的收敛曲线

Fig. 3 Convergence curves of algorithms on test functions

由图 3 可知，在单峰函数 F_1-F_4 上，其他算法对应的收敛曲线均在 SRB-EO 的上方，且都存在陷入局部最优的趋势，说明 SRB-EO 的寻优能力更强，在收敛速度和精度方面存在明显的竞争力。在 F_5 和 F_6 上，SRB-EO 跳出了局部最优值，在单峰函数上的探索能力得到较大提高。在多峰函数 F_7-F_{11} 上，SRB-EO 收敛速度很快，不到 50 代就收敛到理论最优值。在 F_{10} 上，虽然大部分算法不能收敛到理论最优值，但 SRB-EO 能够跳出局部最优，有很好的收敛速度和精度，说明 SRB-EO 在多峰函数上的全局搜索能力更强，收敛精度更高。在固定维多峰函数 F_{13} 上，虽然 SRB-EO 的收敛速度与

大部分算法相当，且在 F_{14} 上稍逊于 HBA 和 UEO，但 SRB-EO 在 F_{12} 上收敛速度最快，在 F_{13} 和 F_{14} 上也优于大部分算法。整体来说，SRB-EO 在固定维多峰函数上在收敛速度和精度方面也有一定的优势。

5.7 Wilcoxon 秩和检验与平均绝对误差检验

为了更加全面地评价 SRB-EO 的性能，本节采用 Wilcoxon 秩和检验方法，以验证 SRB-EO 与其他算法存在显著性差异。若 SRB-EO 与对比算法独立运行 30 次后计算得到 $p \leqslant 0.5$，则两个算法存在显著性差异，反之差异不明显。实验结果如表 7 所列，其中符号“+”“−”和“=”分别表示 SRB-EO 与其他算法存在显著性差异、没有显著性差异和显著性差异相当；NaN 表示需要检验的所有样本数据相同，说明两种算法的性能相当。

由表 7 可知，在 F_7 和 F_9 上，虽然 SRB-EO 与 EO 显著性相当，但是在其他测试函数中 SRB-EO 相比 EO 都存在更优的显著性差异。相比其他智能算法，SRB-EO 在大部分测试函数上都存在显著性差异，因此 SRB-EO 在总体性能方面已经明显优于其他算法。

表 7 Wilcoxon 秩和检验结果（p 值表）

Table 7 Results of Wilcoxon rank-sum test(p value)

函数	HBA		GWO		WOA		EO		MEO		UEO		TLIL-EO	
F_1	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+
F_2	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+
F_3	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+	1.21×10^{-12}	+
F_4	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+
F_5	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+
F_6	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	4.08×10^{-11}	+
F_7	NaN	=	1.17×10^{-12}	+	NaN	=	NaN	=	NaN	=	NaN	=	NaN	=
F_8	NaN	=	1.16×10^{-12}	+	1.89×10^{-6}	+	4.16×10^{-14}	+	NaN	=	0.0055	+	1.69×10^{-14}	+
F_9	NaN	=	0.0419	+	0.3337	−	NaN	=	NaN	=	NaN	=	NaN	=
F_{10}	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	1.07×10^{-9}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	1.29×10^{-9}	+
F_{11}	3.02×10^{-11}	+	3.02×10^{-11}	+	3.02×10^{-11}	+	1.21×10^{-10}	+	3.02×10^{-11}	+	1.21×10^{-12}	+	9.91×10^{-11}	+
F_{12}	0.3749	−	0.0078	+	4.79×10^{-7}	+	0.0207	+	1.78×10^{-10}	+	3.02×10^{-11}	+	1.08×10^{-10}	+
F_{13}	1.04×10^{-4}	+	3.31×10^{-4}	+	0.0256	+	8.80×10^{-4}	+	0.1278	−	1.32×10^{-4}	+	9.46×10^{-8}	+
F_{14}	1.16×10^{-12}	+	1.20×10^{-7}	+	5.39×10^{-10}	+	0.0015	+	2.92×10^{-11}	+	2.64×10^{-4}	+	1.16×10^{-12}	+

平均绝对误差（Mean Absolute Error，MAE）是一种有效评估算法准确率的方式。MAE 值运算公式如下：

$$MAE=\frac{\sum_{i=1}^{N}|M_i-F_i|}{CF} \tag{27}$$

其中，M_i 为实验得到的平均结果，F_i 和 CF 分别为测试函数的理论最优值和个数。MAE 可以避免因算法实验结果与理论最优值相差过大而抵消误差，并能获得用于评估算法的整体误差。基于表 2 中的测试函数，8 种算法的 MAE 如表 8 所列。

表 8 算法的 MAE 值

Table 8 MAE values of each algorithm

算法	MAE	排名	算法	MAE	排名
GWO	2.25	6	UEO	1.84	3
WOA	3.59×10^{3}	8	MEO	2.60	7
HBA	1.84	5	TLIL-EO	1.71	2
EO	1.91	4	SRB-EO	$\mathbf{9.51\times10^{-2}}$	**1**

由表 8 可知，SRB-EO 的 MAE 值最小，说明 SRB-EO 求解问题时的准确率最高，进一步验证了 SRB-EO 拥有更好的勘探能力和收敛精度。以上实验表明，SRB-EO 与其他智能算法相比存在显著性差异，在测试函数上的整体偏差更小，求解精度更高，有着很强的竞争力。

5.8 在高维函数上算法的对比实验

一般算法在求解高维问题时收敛精度和稳定性会有所下降，因此高维函数可以很好地检验算法的收敛精度和鲁棒性。本文选择表 2 中的部分测试函数 F_1-F_{11} 进行 200 维和 500 维实验，用 SRB-EO 与其他 7 种算法进行对比，参数设置与表 1 一致，实验中种群规模设为 30，迭代次数设为 500，独立运行 30 次，记录平均值和标准差。200 维和 500 维的实验结果分别如表 9 和表 10 所列，最优结果加粗标出，其中“Inf”表示无穷大的值，此时算法的收敛效果不好。

表 9 算法在 200 维函数上的实验结果

Table 9 Experimental results of algorithms on 200 dimensional functions

函数	指标	GWO	WOA	HBA	EO	UEO	MEO	TLIL-EO	SRB-EO
F_1	平均值	1.14×10^{-7}	2.27×10^{-70}	1.52×10^{-117}	2.47×10^{-25}	5.84×10^{5}	3.03×10^{-172}	1.59×10^{-296}	**0.00**
	标准差	6.17×10^{-8}	1.24×10^{-69}	5.61×10^{-117}	3.88×10^{-25}	1.99×10^{4}	0.00	0.00	**0.00**
F_2	平均值	2.83×10^{-4}	3.94×10^{-48}	2.39×10^{-62}	1.29×10^{-14}	1.57×10^{305}	1.27×10^{-94}	1.39×10^{-147}	$\mathbf{2.37\times10^{-223}}$
	标准差	8.36×10^{-5}	1.55×10^{-47}	5.48×10^{-62}	1.08×10^{-14}	Inf	5.01×10^{-94}	2.07×10^{-148}	**0.00**
F_3	平均值	1.95×10^{4}	5.35×10^{6}	1.28×10^{-66}	2.13×10^{3}	3.02×10^{-16}	3.69×10^{-145}	1.82×10^{-295}	**0.00**
	标准差	7.95×10^{3}	2.00×10^{6}	7.00×10^{-66}	5.03×10^{3}	1.65×10^{-15}	1.85×10^{-144}	0.00	**0.00**
F_4	平均值	2.44×10^{1}	7.62×10^{1}	3.12×10^{-33}	2.58×10^{1}	2.19×10^{-36}	5.97×10^{-61}	2.50×10^{-149}	$\mathbf{4.74\times10^{-209}}$
	标准差	6.45	2.23×10^{1}	3.97×10^{-33}	2.66×10^{1}	8.33×10^{-36}	1.63×10^{-60}	8.58×10^{-151}	**0.00**
F_5	平均值	1.98×10^{2}	1.98×10^{2}	1.98×10^{2}	1.97×10^{2}	1.98×10^{2}	1.99×10^{2}	1.97×10^{2}	$\mathbf{7.07\times10^{-5}}$
	标准差	3.18×10^{-1}	1.97×10^{-1}	6.05×10^{-1}	7.19×10^{-1}	8.68×10^{-1}	8.52×10^{-2}	9.85×10^{-1}	$\mathbf{2.74\times10^{-4}}$

（续表）

函数	指标	GWO	WOA	HBA	EO	UEO	MEO	TLIL-EO	SRB-EO
F_6	平均值	2.89×10^{1}	1.07×10^{1}	2.86×10^{1}	2.09×10^{1}	2.62×10^{1}	4.82×10^{1}	1.82×10^{1}	**8.43×10^{-8}**
	标准差	1.09	2.65	1.66	1.28	1.76	4.00×10^{-1}	1.23	**2.56×10^{-7}**
F_7	平均值	2.04×10^{1}	7.58×10^{-15}	**0.00**	7.58×10^{-15}	**0.00**	**0.00**	**0.00**	**0.00**
	标准差	1.09×10^{1}	4.15×10^{-14}	**0.00**	4.15×10^{-14}	**0.00**	**0.00**	**0.00**	**0.00**
F_8	平均值	2.39×10^{-5}	5.03×10^{-15}	4.64	8.70×10^{-14}	3.14×10^{-15}	**8.88×10^{-16}**	4.44×10^{-15}	**8.88×10^{-16}**
	标准差	5.79×10^{-6}	2.30×10^{-15}	8.55	2.23×10^{-14}	1.74×10^{-15}	**0.00**	0.00	**0.00**
F_9	平均值	8.74×10^{-3}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	标准差	1.66×10^{-2}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
F_{10}	平均值	5.38×10^{-1}	7.99×10^{-2}	4.01×10^{-1}	1.98×10^{-1}	3.05×10^{-1}	1.15	1.57×10^{-1}	**2.99×10^{-9}**
	标准差	5.80×10^{-2}	3.78×10^{-2}	5.34×10^{-2}	2.74×10^{-2}	5.17×10^{-2}	2.54×10^{-2}	2.02×10^{-2}	**1.08×10^{-8}**
F_{11}	平均值	1.68×10^{1}	5.90	1.90×10^{1}	1.82×10^{1}	1.97×10^{1}	1.99×10^{1}	1.93×10^{1}	**4.81×10^{-8}**
	标准差	4.22×10^{-1}	1.70	4.07×10^{-1}	5.55×10^{-1}	1.71×10^{-1}	3.50×10^{-2}	4.96×10^{-1}	**1.29×10^{-7}**

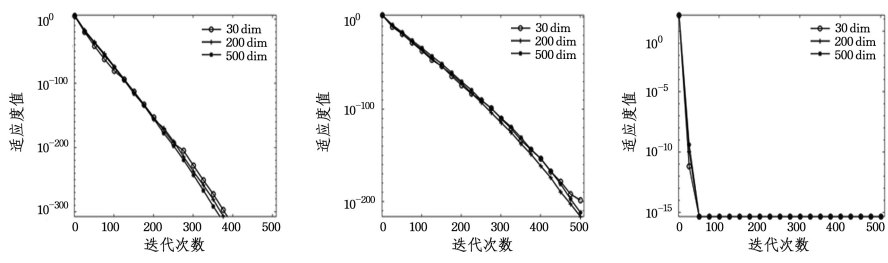
表 10 算法在 500 维函数上的实验结果

Table 10 Experimental results of algorithms on 500 dimensional functions

函数	指标	GWO	WOA	HBA	EO	UEO	MEO	TLIL-EO	SRB-EO
F_1	平均值	1.71×10^{-3}	5.67×10^{-67}	1.49×10^{-112}	8.78×10^{-23}	1.53×10^{6}	1.81×10^{-149}	4.43×10^{-296}	**0.00**
	标准差	6.59×10^{-4}	3.10×10^{-66}	$7.93\times10^{-11}2$	7.66×10^{-23}	3.30×10^{4}	9.89×10^{-149}	0.00	**0.00**
F_2	平均值	1.48×10^{-9}	3.52×10^{-48}	Inf	2.38×10^{-13}	1.75×10^{306}	1.93×10^{-85}	3.05×10^{-147}	**7.30×10^{-234}**
	标准差	5.62×10^{-9}	1.79×10^{-47}	Inf	2.08×10^{-13}	Inf	9.40×10^{-85}	5.22×10^{-148}	**0.00**
F_3	平均值	3.09×10^{5}	3.02×10^{7}	2.53×10^{-62}	2.98×10^{4}	3.78×10^{7}	1.67×10^{-128}	1.15×10^{-294}	**0.00**
	标准差	7.09×10^{4}	1.02×10^{7}	1.38×10^{-61}	3.34×10^{4}	1.32×10^{7}	5.92×10^{-128}	0.00	**0.00**
F_4	平均值	6.54×10^{1}	8.06×10^{1}	1.26×10^{-28}	7.41×10^{1}	9.91×10^{1}	5.37×10^{-41}	2.68×10^{-149}	**1.36×10^{-210}**
	标准差	4.76	2.29×10^{1}	1.72×10^{-28}	2.09×10^{1}	3.42×10^{-1}	2.47×10^{-40}	5.75×10^{-151}	**0.00**
F_5	平均值	4.98×10^{2}	4.96×10^{2}	4.98×10^{2}	4.98×10^{2}	7.17×10^{9}	4.99×10^{2}	4.97×10^{2}	**4.41×10^{-4}**
	标准差	3.78×10^{-1}	3.91×10^{-1}	3.24×10^{-1}	2.05×10^{-1}	2.32×10^{8}	3.74×10^{-2}	5.26×10^{-1}	**1.35×10^{-3}**
F_6	平均值	9.16×10^{1}	3.18×10^{1}	9.84×10^{1}	8.71×10^{1}	1.52×10^{6}	1.24×10^{2}	8.15×10^{1}	**8.02×10^{-7}**
	标准差	1.82	8.33	1.87	1.64	3.16×10^{4}	2.36×10^{-1}	2.36	**1.82×10^{-6}**
F_7	平均值	7.25×10^{1}	6.06×10^{-14}	**0.00**	1.21×10^{-13}	**0.00**	**0.00**	**0.00**	**0.00**
	标准差	3.38×10^{1}	3.32×10^{-13}	**0.00**	3.14×10^{-13}	**0.00**	**0.00**	**0.00**	**0.00**
F_8	平均值	-5.67×10^{4}	-1.75×10^{5}	-7.16×10^{4}	-7.44×10^{4}	-8.97×10^{3}	-2.07×10^{4}	4.44×10^{-15}	**8.88×10^{-16}**
	标准差	2.80×10^{3}	2.84×10^{4}	7.88×10^{3}	5.14×10^{3}	1.90×10^{3}	2.44×10^{3}	0.00	**0.00**
F_9	平均值	1.33×10^{-2}	**0.00**	**0.00**	7.77×10^{-17}	**0.00**	**0.00**	**0.00**	**0.00**
	标准差	3.03×10^{-2}	**0.00**	**0.00**	5.17×10^{-17}	**0.00**	**0.00**	**0.00**	**0.00**
F_{10}	平均值	7.71×10^{-1}	1.00×10^{-1}	7.51×10^{-1}	5.83×10^{-1}	1.76×10^{10}	1.19	5.00×10^{-1}	**9.53×10^{-10}**
	标准差	7.36×10^{-2}	5.30×10^{-2}	2.74×10^{-2}	2.68×10^{-2}	7.06×10^{8}	6.49×10^{-3}	3.07×10^{-2}	**1.86×10^{-9}**
F_{11}	平均值	5.10×10^{1}	1.72×10^{1}	4.94×10^{1}	4.92×10^{1}	3.16×10^{10}	4.99×10^{1}	4.96×10^{1}	**1.73×10^{-7}**
	标准差	1.62	6.92	2.15×10^{-1}	1.89×10^{-1}	1.06×10^{9}	3.70×10^{-2}	1.35×10^{-1}	**4.25×10^{-7}**

由表 9 和表 10 可知，UEO 在 F_1 和 F_2 上偏离理论最优值，该算法在面对高维实验时稍有些无力。HBA 在 500 维的 F_2 上的平均值和标准差为无穷大值，产生较大偏差。EO 在 F_3 和 F_4 上都陷入局部最优值，且在 F_7 和 F_9 上不如 UEO，MEO，TLIL-EO 和 SRB-EO 这些改进算法的收敛效果好。EO，UEO 和 MEO 在 500 维的 F_8 上均值偏离最优解，虽然 TLIL-EO 的收敛效果比以上算法好，但是收敛精度不如 SRB-EO，在 F_5，F_6，F_{10}，F_{11} 上只有 SRB-EO 能够避免局部停滞，达到理论最优值。

图 4 为 SRB-EO 在部分不同维数函数上的收敛曲线图。由图 4 可知，在不同维数的测试函数上，SRB-EO 的收敛曲线图无较大差异，并且都能收敛到最优值，说明维数的增加基本不会改变 SRB-EO 的寻优性能。综合来看，在高维问题上，SRB-EO 有着更好的可扩展性、收敛精度和鲁棒性。



(a)F_1 (b)F_4 (c)F_8

图 4 不同维数下 SRB-EO 的收敛曲线

Fig. 4 Convergence curves of SRB-EO with different dimensions

6 工程设计问题

工程设计是满足产品开发需求的过程。压力容器设计和三杆桁架问题是典型的工程设计问题，两个工程设计模型图如图 5 和图 6 所示。为了进一步检验算法解决实际问题的性能，应用 SRB-EO 与其他算法来解决工程设计问题。为保证

实验的公平性，实验中种群数量为 30，迭代次数为 500。

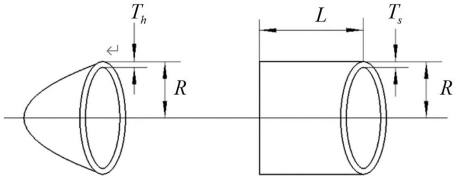


图 5 压力容器设计模型图

Fig. 5 Model diagram of pressure vessel design

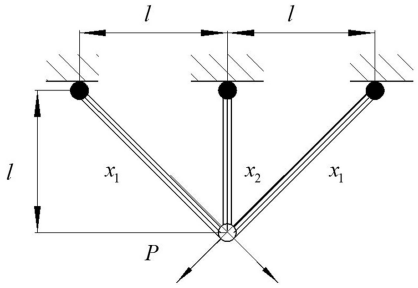


图 6 三杆桁架问题模型图

Fig. 6 Model diagram of three-bar truss design

6.1 压力容器设计

压力容器设计是最流行的工程应用之一，其目的是将压力容器的焊接、制造和材料成本降至最低。该问题由 4 个变量和 4 个约束条件组成。变量包括壳体厚度 $T_s(x_1)$、封头厚度 $T_h(x_2)$、内径 $R(x_3)$和容器圆柱形截面长度 $L(x_4)$。数学公式定义如下：

$$f(x)=0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3$$

$$\begin{cases} g_1(x)=-x_1+0.0193x_3\leqslant 0 \\ g_2(x)=-x_2+0.0954x_3\leqslant 0 \\ g_3(x)=-\pi x_3^2x_4-\dfrac{4}{3}\pi x_3^3+1\,296\,000\leqslant 0 \\ g_4(x)=x_4-240\leqslant 0 \\ 0\leqslant x_1,x_2\leqslant 99 \\ 10\leqslant x_3,x_4\leqslant 200 \end{cases} \tag{28}$$

表 11 列出了 5 种算法解决压力容器设计问题的最优结果。由表 11 可知，SRB-EO 找到的最优成本比原始 EO 要少，也比最新的算法 m-EO 和 HAOA 更好，表明 SRB-EO 在解决压力容器设计问题时更具优势。

表 11 压力容器设计问题的实验结果

Table 11 Experimental results of pressure vessel design

算法	$T_s(x_1)$	$T_h(x_2)$	$R(x_3)$	$L(x_4)$	结果
PSC-EO	**0.7781**	**0.3846**	**40.3196**	200.000	**5885.333**
EO[7]	0.8125	0.4375	42.0977	176.6472	6059.836
m-EO[28]	0.8125	0.4375	42.0984	176.6366	6059.714
GWO[21]	0.8125	0.4345	42.0892	176.7587	6015.639
HAOA[29]	0.8580	0.4204	46.0400	**133.2321**	5972.027

6.2 三杆桁架问题

三杆桁架问题的目的是找到钢筋横截面积的最优值和最小化结构重量，同时使 3 根钢筋上的应力约束最大化。该问题存在一个具有 3 个非线性不等式约束的适应度函数，并且连续变量(x_1,x_2)用于决策。数学模型可以表示如下：

$$f(x)=(2\sqrt{2}x_1+x_2)\times l$$

$$\begin{cases} g_1(x)=\dfrac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\leqslant 0 \\ g_2(x)=\dfrac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\leqslant 0 \\ g_3(x)=\dfrac{1}{\sqrt{2}x_2+x_1}P-\sigma\leqslant 0 \\ 0\leqslant x_1\leqslant 1,0\leqslant x_2\leqslant 1 \end{cases} \tag{29}$$

其中 $l=100\,\text{cm}$，$P=2\,\text{KN/cm}^2$，$\sigma=2\,\text{KN/cm}^2$。

表 12 列出了 5 种算法解决三杆桁架问题时找到的最优结果。由表 12 可知，SRB-EO 比原始 EO 和 m-EO 寻优效果好，不管是经典的 GWO 和最新的 SFO，改进算法都有一定的优势，验证了 SRB-EO 在解决三杆桁架问题时的出色求解能力。

表 12 三杆桁架问题的实验结果

Table 12 Experimental results of three-bar truss design

算法	x_1	x_2	结果
SRB-EO	**0.787160**	**0.408240**	**263.4665**
EO[7]	0.788030	0.410070	263.8961
m-EO[28]	0.788510	0.408730	263.8959
GWO[21]	0.787720	0.410960	263.8977
SFO[3]	0.788456	0.408868	263.8959

结束语 本文提出了一种融合麻雀搜索和随机差分的双向学习平衡优化器算法(SRB-EO)。该算法利用自适应种群划分策略来平衡 EO 的探索和开发能力，从而提高了算法的整体寻优精度和鲁棒性；算法中的随机差分平衡池策略有利于增加个体间信息的交流，从而增强了算法的全局寻优能力；双向混沌反向学习策略丰富了种群多样性，并使算法的寻优精度得到进一步提升。SRB-EO 比其他智能算法表现出更优的收敛精度、收敛速度、鲁棒性和可扩展性。两个工程设计问题验证了 SRB-EO 的有效性和实用性。

本文仅对 EO 的性能进行改进，未来可结合具体现实背景，将改进算法应用到作业车间调度、多目标优化和车辆路径等具体问题中，并进一步提高算法性能。

参 考 文 献

[1] JAIN M, SINGH V, RANI A. A novel nature-inspired algorithm for optimization: Squirrel search algorithm[J]. Swarm and Evolutionary Computation, 2019, 44: 148-175.

[2] HEIDARI A, MIRJALILI S, FARIS H, et al. Harris hawks optimization: Algorithm and applications[J]. Future Generation Computer Systems, 2019, 97: 849-872.

[3] SHADRAVAN S, NAJI H R, BARDSIRI V K. The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems[J]. Engineering Applications of Artificial Intelligence, 2019, 80: 20-34.

[4] TIAN Y, ZHANG X, WANG C, et al. An evolutionary algorithm for large-scale sparse multiobjective optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2019, 24(2): 380-393.

[5] CAO L, XU L, GOODMAN E D, et al. Evolutionary dynamic multiobjective optimization assisted by a support vector regres-

sion predictor[J]. IEEE Transactions on Evolutionary Computation, 2019, 24(2): 305-319.
[6] XU M, JIAO J J, LONG W, et al. Sine cosine algorithm based on logistic model and stochastic differential mutation[J]. Computer Science, 2020, 47(2): 206-212.
[7] FARAMARZI A, HEIDARINEJAD M, STEPHENS B, et al. Equilibrium optimizer: A novel optimization algorithm [J]. Knowledge-Based Systems, 2020, 191: 105190.
[8] ABDEL-BASSET M, MOHAMED R, MIRJALILI S. A binary equilibrium optimization algorithm for 0-1 knapsack problems [J]. Computers & Industrial Engineering, 2021, 151: 106946.
[9] GAO Y, ZHOU Y, LUO Q. An efficient binary equilibrium optimizer algorithm for feature selection[J]. IEEE Access, 2020, 8: 140936-140963.
[10] HOUSSEIN E H, DIRAR M, ABUALIGAH L, et al. An efficient equilibrium optimizer with support vector regression for stock market prediction [J]. Neural Computing and Applications, 2022, 34(4): 3165-3200.
[11] WANG J, YANG B, LI D, et al. Photovoltaic cell parameter estimation based on improved equilibrium optimizer algorithm[J]. Energy Conversion and Management, 2021, 236(3): 114051.
[12] HOUSSEIN E H, HELMY B E, OLIVA D, et al. An efficient multi-thresholding based COVID-19 CT images segmentation approach using an improved equilibrium optimizer[J]. Biomedical Signal Processing and Control, 2022, 73: 103401.
[13] WOLPERT D H, MACREADY W G. No free lunch theorems for optimization[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67-82.
[14] HEMALATHA R, UMAMAHESWARI R, JOTHI S. Optimal route maintenance based on adaptive equilibrium optimization and GTA based route discovery model in MANET[J]. Peer-to-Peer Networking and Applications, 2021, 14(6): 3416-3430.
[15] OUADFEL S, ABD ELAZIZ M. Efficient high-dimension feature selection based on enhanced equilibrium optimizer[J]. Expert Systems with Applications, 2022, 187: 115882.
[16] JIA H, PENG X. High equilibrium optimizer for global optimization[J]. Journal of Intelligent & Fuzzy Systems, 2021, 40(3): 5583-5594.
[17] XUE J, SHEN B. A novel swarm intelligence optimization approach: sparrow search algorithm[J]. Systems Science & Control Engineering, 2020, 8(1): 22-34.
[18] LIU J, LAMPINEN J. A fuzzy adaptive differential evolution algorithm[J]. Soft Computing, 2005, 9(6): 448-462.
[19] YILDIZ B S, PHOLDEE N, BUREERAT S, et al. Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems[J]. Engineering with Computers, 2022, 38(5): 4207-4219.
[20] YANG D, LIU Z, ZHOU J. Chaos optimization algorithms based on chaotic maps with different probability distribution and search speed for global optimization [J]. Communications in Nonlinear Science & Numerical Simulation, 2014, 19(4): 1229-1246.
[21] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69: 46-61.
[22] MIRJALILI S, LEWIS A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.
[23] HASHIM F A, HOUSSEIN E H, HUSSAIN K, et al. Honey badger algorithm: New metaheuristic algorithm for solving optimization problems[J]. Mathematics and Computers in Simulation, 2022, 192: 84-110.
[24] GUI P, HE F, LING B W K, et al. United equilibrium optimizer for solving multimodal image registration[J]. Knowledge-Based Systems, 2021, 233: 107552.
[25] MINOCHA S, SINGH B. A novel equilibrium optimizer based on Lévy flight and iterative cosine operator for engineering optimization problems[J]. Expert Systems, 2022, 39(2): e12843.
[26] ZHOU P, DONG C Y, CHEN X Y, et al. An equilibrium optimizer algorithm based on a tent chaos and lens imaging learning strategy[J/OL]. Control and Decision: 1-9. [2022-12-06]. DOI: 10.13195/j.kzyjc.2021.1537.
[27] YAO X, LIU Y, LIN G. Evolutionary programming made faster [J]. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82-102.
[28] GUPTA S, DEEP K, MIRJALILI S. An efficient equilibrium optimizer with mutation strategy for numerical optimization[J]. Applied Soft Computing, 2020, 96: 106542.
[29] JIA H M, LIU Y X, LIU Q X, et al. Hybrid algorithm of slime mould algorithm and arithmetic optimization algorithm based on random opposition-based learning [J]. Journal of Frontiers of Computer Science and Technology, 2022, 16(5): 1182-1192.

HOU Xinyu, born in 1998, postgraduate, is a member of China Computer Federation. His main research interest is optimization and control.

LU Haiyan, born in 1970, Ph. D, associate professor, master supervisor. Her main research interests include combination optimization and intelligent algorithms.

(责任编辑:柯颖)